ESAIM: Control, Optimisation and Calculus of Variations Will be set by the publisher
URL: http://www.emath.fr/cocv/

A SET ORIENTED APPROACH TO GLOBAL OPTIMAL CONTROL*

Oliver Junge[1] and Hinke M Osinga[2]

Abstract. We describe an algorithm for computing the value function for "all source, single destination" discrete-time nonlinear optimal control problems together with approximations of associated globally optimal control strategies. The method is based on a set oriented approach for the discretization of the problem in combination with graph-theoretic techniques. The central idea is that a discretization of phase space of the given problem leads to an (all source, single destination) *shortest path problem* on a finite graph. The method is illustrated by two numerical examples, namely a single pendulum on a cart and a parametrically driven inverted double pendulum.

1991 Mathematics Subject Classification. 49M25,49J53,65K10,90C39.

December 4, 2002.

Contents

Keywords and phrases: global optimal control, value function, set oriented method, shortest path

* *We would like to thank Robert Preis for providing an implementation of Dijkstra's algorithm for* GAIO. *We gratefully acknowledge helpful discussions with Lars Grüne and Alexander Vladimirksy on the contents of this paper. HMO thanks the University of Paderborn for their hospitality and financial support during her visit.*

[1] Institute for Mathematics, University of Paderborn, 33095 Paderborn, Germany, junge@upb.de
[2] Engineering Mathematics, University of Bristol, Bristol BS8 1TR, UK, H.M.Osinga@bristol.ac.uk

1. Introduction

The idea to solve optimal control problems by searching for shortest paths is immediately clear for specific classes of control problems. For example, consider the problem of moving a particle from a given position to a desired destination, subject to a cost function that is directly related to the arclength of the path followed, or let the cost function be simply time itself so that the shortest path is defined as the fastest path. In this paper we investigate how to rephrase more general optimal control problems in terms of finding a shortest path. The main idea is to consider a discrete-time version of the model which is translated into a graph-theoretic description. The problem then becomes that of finding a path of minimal length from the initial state to the final state in a directed weighted graph. We show how to construct a finite graph so that standard shortest path algorithms from graph theory (like *Dijkstra's* algorithm [5]) can immediately be applied in order to solve the problem. The advantage of these techniques is that the approximate optimal cost and approximate optimal trajectories for *all* possible initial points are calculated simultaneously.

We emphasize that it is possible to use this graph-theoretic approach when the state space is $\mathbb{R}^d$, that is, it contains an uncountable number of points. Via a multilevel discretization of phase space one can efficiently derive finite directed weighted graphs that serve as coarse models for the evolution of the underlying control system. On each of these graphs an application of a shortest path algorithm yields an approximation to the (optimal) value function of the problem and to corresponding nearly optimal trajectories.

There have been few attempts to exploit the efficiency of graph-theoretic tools in the context of optimal control. Especially in robotic navigation, the idea of finding shortest paths is used for spaces with stationary or moving obstacles. Here, the emphasis is on finding just one rather than the optimal path. Brooks and Lozano-Pérez [1] present a subdivision algorithm for finding such collision-free paths. The essence of their work lies in constructing an appropriate partition of the configuration space and finding a string of connected cells. This construction can be extremely complicated due to the geometry of the obstacles and of the moving object. The

aim is to find *one* path, and the constructed graph, with cells as vertices and edges indicating neighboring cells, is not complete.

More recently, Tsitsiklis [18] described a graph-theoretic approach to a problem of finding an *optimal* trajectory; see also [13]. The moving object is viewed as a particle and any obstacles are stationary. The aim is to minimize the trajectory until it reaches the boundary of a pre-specified subset of $\mathbb{R}^d$, at which a terminal cost is incurred. Hence, if this boundary is one particular point, namely the desired destination, the problem is similar to what is considered in this paper. Obstacles are avoided by imposing an infinite terminal cost at the boundaries surrounding them, where it is assumed that the obstacles admit a finite description. The construction of the associated finite graph is based on a rather ad-hoc discretization and the method is, unfortunately, restricted to problems where the cost function does not explicitly depend on the control.

Standard approaches to the solution of problems of the type considered in this paper include (adaptive) finite difference schemes for the solution of the (discrete) Hamilton-Jacobi-Bellmann equation, see e.g. the work of Falcone [6] and Grüne [8]. More recently, Sethian and Vladimirsky introduced so called *ordered upwind methods* that yield efficient non-iterative schemes.

In this paper, we consider the problem of finding an optimal path from any initial condition to a prescribed final destination subject to an *arbitrary* continuous nonnegative cost function – in particular, the cost function may also depend on the control. We describe how to construct a finite directed weighted graph based on a very natural discretization of the problem, and use standard shortest path algorithms in order to find the solution. The computed shortest path yields a *pseudo-trajectory* of the underlying discrete-time control system, which should then be employed as an initial guess for standard (local) solvers in order to find a true trajectory of the underlying system. We do not yet consider the presence of obstacles, but are confident that the algorithm can be modified by removing vertices associated with the obstacles from the graph.

A more detailed outline of the paper is as follows: we start with a description of the discrete-time optimal control problem under consideration in Section 2. Section 3 contains the main

algorithm together with a statement about its convergence. In Section 4 we outline the implementational aspects of our approach. Finally in Section 5 we illustrate the performance of the method by two numerical examples: a single and a double inverted pendulum.

2. Problem formulation

Consider the discrete-time control system

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, \tag{1}$$

with $f : X \times U \to \mathbb{R}^d$ continuous. Here, $X \subset \mathbb{R}^d$ is a particular (compact) region of interest, and the set $U \subset \mathbb{R}^m$ of admissible controls is compact. We assume that the origin of the uncontrolled system is a fixed point, that is, $f(0, 0) = 0$. Our goal is, starting at some point $x \in X$, to impose controls $\mathbf{u} = (u_0, u_1, \ldots) \in U^{\mathbb{N}}$ such that the trajectory $(x_k(x, \mathbf{u}))_{k\in\mathbb{N}}$, defined as

$$x_0(x, \mathbf{u}) = x \quad \text{and} \quad x_{k+1}(x, \mathbf{u}) = f(x_k(x, \mathbf{u}), u_k), \quad k = 0, 1, \ldots$$

remains in X for all $k \geq 0$ and satisfies $x_k(x, \mathbf{u}) \to 0$ as $k \to \infty$ — if possible. Evidently, there may be $x \in X$ for which there is no control $u \in U$ such that $f(x, u) \in X$ or for which no "stabilizing" control sequence $\mathbf{u}$ exists. At each step of this iteration process we incur an instantaneous (nonnegative) cost $q(x, u)$, where

$$q : X \times U \to [0, \infty)$$

is continuous and satisfies $q(0, 0) = 0$. In addition to stabilizing our system we aim to minimize the total cost

$$J(x, \mathbf{u}) = \sum_{k=0}^{\infty} q(x_k(x, \mathbf{u}), u_k) \in [0, \infty]$$

that we accumulate along a trajectory. To this end, for every x, we let

$$\mathcal{U}(x) = \{\mathbf{u} \in U^{\mathbb{N}} : x_k(x, \mathbf{u}) \to 0 \text{ as } k \to \infty\}$$

be the set of stabilizing control sequences for x. The set S of all $x \in X$ for which $\mathcal{U}(x)$ is nonempty is called the *stabilizable subset*.

Our goal is to compute an approximation of the *(optimal) value function*

$$V(x) = \inf_{\mathbf{u}\in\mathcal{U}(\mathbf{x})} J(x, \mathbf{u}).$$

We let $\inf \emptyset = \infty$, i.e. we set $V(x) = \infty$ for x with $\mathcal{U}(x) = \emptyset$, that is, for $x \notin S$. It turns out that our computed approximation is always finite for $x \in S$. As part of the computation, we obtain an approximation for the stabilizable subset S and also an approximate optimizing sequence $\mathbf{u} = \mathbf{u}(x)$ for every $x \in S$.

The linear-quadratic case

Although the method proposed in this paper works for general (nonlinear) control systems and cost functions (we only assume both of them to be continuous) we briefly recall the classic setting of linear-quadratic problems at this point:

In general, it is not possible to find the value function V analytically. Only for the class of so-called *linear-quadratic* (LQ) optimal control problems V can be found explicitly. LQ problems are of the form

$$x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \ldots,$$

with associated cost function

$$q(x, u) = x^T Qx + u^T Ru,$$

where $A \in \mathbb{R}^{d\times d}$, $B \in \mathbb{R}^{d\times r}$, $Q \in \mathbb{R}^{d\times d}$, and $R \in \mathbb{R}^{r\times r}$. The pair (A, B) must be stabilizable, that is, a matrix $F \in \mathbb{R}^{r\times d}$ exists such that $A + BF$ has all its eigenvalues inside the unit circle. Furthermore, Q must be positive definite and R positive semi-definite.

For this class of problems the value function is $V(x) = x^*Px$, where $P \in \mathbb{R}^{d\times d}$ is the unique positive definite matrix that satisfies the discrete-time algebraic Riccati equation

$$P + A^T PB(B^T PB + R)^{-1}B^T PA - A^T PA - Q = 0.$$

The optimal controls are then obtained by the feedback law

$$u = -(B^T PB + R)^{-1}B^T PAx.$$

We refer to [16] for more details.

3. Computing the value function

Bellman's principle of optimality (see e.g. [16]) states that for all x such that $V(x)$ is finite we have the following recursive relation

$$V(x) = \inf_{u \in U} \{q(x,u) + V(f(x,u))\}. \tag{2}$$

Hence, if we would know the value of V in a neighbourhood of the origin, then we could extend this known domain to a larger neighbourhood using the above with V as the "cost-to-go." This construction is reminiscent of algorithms for computing *shortest paths* in graph theory.

3.1. **Graph-theoretic approach**

We can represent the evolution of Equation (1) as a directed graph

$$G = (X, E), \quad E = \{(x, f(x,u)) \mid x \in X, f(x,u) \in X, u \in U\},$$

with infinitely many nodes $x \in X$ and infinitely many edges $e \in E$. The cost q of the evolution is represented as a weight on the edges. Each edge $e = (x, f(x,u))$ carries the weight $w(e) = q(x,u) \geq 0$. Then, roughly speaking, for all x the value function $V(x)$ is the infimum over the length of all paths connecting x to 0 in the graph G.

In order to obtain an algorithm for the computation of V we are going to approximate G by a finite graph. This is the reason for restricting ourselves to a bounded subset $X \subset \mathbb{R}^d$ and why we can only consider controlled trajectories that remain in X. To this end we consider a *partition* $\mathcal{P}$ of X, that is, a finite collection of compact subsets $P_i \subset X$ with $\cup_i P_i = X$, and $m(P_i \cap P_j) = 0$ (where m denotes Lebesgue measure). We can now define a finite approximation to G, namely the graph

$$G_{\mathcal{P}} = (\mathcal{P}, E_{\mathcal{P}}), \quad E_{\mathcal{P}} = \{(P_i, P_j) \in \mathcal{P} \times \mathcal{P} \mid f(P_i, U) \cap P_j \neq \emptyset\},$$

where the edge $e = (P_i, P_j)$ carries the weight

$$w(e) = \min_{x \in P_i, u \in U} \{q(x,u) \mid f(x,u) \in P_j\}.$$

Note that, due to the compactness of P_i, P_j and U, and the continuity of f and q, we indeed have a minimum here. Of course determining $E_{\mathcal{P}}$ together with the weights $w(e)$, $e \in E_{\mathcal{P}}$, is the crucial computational part of the whole approach. We refer to Section 4 for a description of how these computations can be carried out in an efficient way. Let us emphasize again that $G_{\mathcal{P}}$ is a graph with finitely many nodes.

We use $G_{\mathcal{P}}$ to find an approximation of the value function V. For any $x \in X$ there is a subset $P(x) \in \mathcal{P}$ containing x. The approximation for $V(x)$ will be the length of the *shortest path* from a node $P(x)$ to a node $P(0) \in \mathcal{P}$ that contains the origin. The length of a path is determined as follows. Let $p = (e_1, \dots, e_m)$, $e_k \in E_{\mathcal{P}}$, be a *path* in $G_{\mathcal{P}}$. The *length* of p is defined as

$$w(p) = \sum_{k=1}^{m} w(e_k).$$

We say that a path $p = p(x) = (e_1, \dots, e_m), e_k \in E_{\mathcal{P}}$ in $G_{\mathcal{P}}$ *connects* x to 0, if $e_1 = (P(x), P_1)$ and $e_m = (P_{m-1}, P(0))$ for some sets $P_1, P_{m-1} \in \mathcal{P}$. If no such path exists then x is not stabilizable (in fact, none of the points in $P(x)$ is) and $V(x) = \infty$. For all $x \in X$ we approximate $V(x)$ by

$$V_{\mathcal{P}}(x) = \min\{w(p(x)) \mid p(x) \text{ connects } x \text{ to } 0\}, \tag{3}$$

where, again, we set $\min \emptyset = \infty$. One should realize that an edge $e = (P_i, P_j)$ exists as soon as the image of the set P_i for some control $u \in U$ intersects the set P_j. Note that for a given path $p(x) = (e_1, \dots, e_m), e_i = (P_{i-1}, P_i)$ there need not exist a trajectory $(x_k(x, \mathbf{u}))_{k \in \mathbb{N}}$ such that $x_k(x, \mathbf{u}) \in P_k$ for $k = 0, \dots, m$. This means that the value of $V_{\mathcal{P}}(x)$ may be less than that of $V(x)$, or we may find a path $p(x)$ connecting x to 0 for some $x \in X$ that is in fact not stabilizable. More formally we have the following property for $V_{\mathcal{P}}$:

Proposition 3.1. *For every partition $\mathcal{P}$, $V_{\mathcal{P}}$ is a lower bound on V.*

Proof. The statement obviously holds for $x \in X$ with $V(x) = \infty$. Hence, consider $x \in X$ with $V(x) < \infty$. It suffices to show that, for arbitrary $\varepsilon > 0$, there is a path $p(x)$ in $G_{\mathcal{P}}$ connecting x to 0 such that $w(p(x)) \leq J(x, \mathbf{u})$, where $\mathbf{u} = (u_0, u_1, \dots) \in \mathcal{U}(x)$ is a sequence of controls such that $J(x, \mathbf{u}) < V(x) + \varepsilon$.

Let $p(x)$ be a path in $G_{\mathcal{P}}$ that is defined by following the trajectory $(x_k(x,\mathbf{u}))_{k\in\mathbb{N}}$, namely

$$p(x) = (e_1, \dots, e_m), \quad e_k = (P(x_{k-1}), P(x_k)), \quad k = 1, \dots, m,$$

where m is such that $x_m \in P(0)$. The length of this path is

$$\begin{aligned} w(p(x)) &= \sum_{k=1}^{m} w(e_k) = \sum_{k=1}^{m} \min_{x\in P(x_{k-1}), u\in U} \{q(x,u) \mid f(x,u) \in P(x_k)\} \\ &\leq \sum_{k=1}^{m} q(x_{k-1}, u_{k-1}) \leq \sum_{k=1}^{\infty} q(x_{k-1}, u_{k-1}) = J(x,\mathbf{u}). \end{aligned}$$

□

Note that there are three factors that make $w(p(x))$ smaller than $V(x)$: (i) The finiteness of the number of edges m, (ii) the computation of the weights $w(e_k)$ by minimizing q, (iii) the fact that the shortest path $p(x)$ may actually pass through a different set of partition elements than a (near) optimal trajectory. All these are a direct result of the definition of $G_{\mathcal{P}}$ and depend on the choice of the partition $\mathcal{P}$. However, in order to deal with (i), one could redefine $V_{\mathcal{P}}$ by adding a final cost for $x_m \in \mathcal{P}(0)$. For example, this final cost function could be defined as the optimal cost to steer x_m to the origin using the linearized equations

$$x_{k+1} = Ax_k + Bu_k, \quad A = D_x f(0,0), \quad B = D_u f(0,0)$$

and quadratic cost

$$x^*Qx + u^*Ru, \quad Q = D_x^2 q(0,0), \quad R = D_u^2 q(0,0).$$

As explained in Section 2, the value function of this linear-quadratic problem is known explicitly. Moreover, in a neighborhood of the origin, this indeed leads to a good approximation of V. Our results hold as long as the final cost on $\mathcal{P}(0)$ is a lower bound for V.

3.2. **Convergence of $V_{\mathcal{P}}$ to V**

We are now going to establish a statement about the convergence of $V_{\mathcal{P}}$ to V as the diameter of $\mathcal{P}$, defined as $\operatorname{diam}(\mathcal{P}) := \max_i \operatorname{diam}(P_i)$, goes to zero. To this end let $(\mathcal{P}^{(\ell)})_{\ell\in\mathbb{N}}$ be a sequence of nested partitions of X with $\operatorname{diam}(\mathcal{P}^{(\ell)}) \to 0$ as $\ell \to \infty$. By "nested" we mean that for all

ℓ and every $P_i^{(\ell+1)} \in \mathcal{P}^{(\ell+1)}$ there is a $P_i^{(\ell)} \in \mathcal{P}^{(\ell)}$ with $P_i^{(\ell+1)} \subset P_i^{(\ell)}$. We have the following observation:

Proposition 3.2. *For $x \in X$ and $\ell \in \mathbb{N}$ let $p^{(\ell)} = p^{(\ell)}(x)$ be a path in $G_{\mathcal{P}^{(\ell)}}$ with minimal length achieving $V_{\mathcal{P}^{(\ell)}}(x)$ (see (3)). Then the sequence $(w(p^{(\ell)}))_{\ell\in\mathbb{N}}$ is monotonically increasing, that is, for all ℓ,*

$$w(p^{(\ell+1)}) \geq w(p^{(\ell)}).$$

Proof. Suppose that for some ℓ there would be minimizing paths $p^{(\ell)}(x)$ in $G_{\mathcal{P}^{(\ell)}}$ and $p^{(\ell+1)}(x)$ in $G_{\mathcal{P}^{(\ell+1)}}$ such that $w(p^{(\ell+1)}(x)) < w(p^{(\ell)}(x))$. Using $p^{(\ell+1)}(x)$ we are going to construct a path $\tilde{p}^{(\ell)}(x)$ in $G_{\mathcal{P}^{(\ell)}}$ with $w(\tilde{p}^{(\ell)}(x)) < w(p^{(\ell)}(x))$, contradicting the minimality of $p^{(\ell)}(x)$.

Let $p^{(\ell+1)}(x) = (e_1^{(\ell+1)}, \dots, e_{m(\ell+1)}^{(\ell+1)})$, with $e_k^{(\ell+1)} = (P_{k-1}^{(\ell+1)}, P_k^{(\ell+1)}) \in E_{\mathcal{P}}^{(\ell+1)}$. Hence, $f(P_{k-1}^{(\ell+1)}, U) \cap P_k^{(\ell+1)} \neq \emptyset$, for all $k = 1, \dots, m(\ell+1)$. Since the partitions $\mathcal{P}^{(\ell)}$ are nested, there are sets $\tilde{P}_k^{(\ell)} \in \mathcal{P}^{(\ell)}$ such that $P_k^{(\ell+1)} \subset \tilde{P}_k^{(\ell)}$ for $k = 0, \dots, m(\ell+1)$. This means that $f(\tilde{P}_{k-1}^{(\ell)}, U) \cap \tilde{P}_k^{(\ell)} \neq \emptyset$, and thus $\tilde{e}_k^{(\ell)} = (\tilde{P}_{k-1}^{(\ell)}, \tilde{P}_k^{(\ell)})$ is an edge in $E_{\mathcal{P}}^{(\ell)}$. Therefore, $\tilde{p}^{(\ell)}(x) = (\tilde{e}_1^{(\ell)}, \dots, \tilde{e}_{m(\ell+1)}^{(\ell)})$ is a path. Furthermore, for all $k = 1, \dots, m(\ell+1)$,

$$\begin{aligned} w(\tilde{e}_k^{(\ell)}) &= \min_{x\in\tilde{P}_{k-1}^{(\ell)}, u\in U} \{q(x,u) \mid f(x,u) \in \tilde{P}_k^{(\ell)}\} \\ &\leq \min_{x\in P_{k-1}^{(\ell+1)}, u\in U} \{q(x,u) \mid f(x,u) \in P_k^{(\ell+1)}\} = w(e_k^{(\ell+1)}). \end{aligned}$$

This yields $w(\tilde{p}^{(\ell)}(x)) \leq w(p^{(\ell+1)}(x)) < w(p^{(\ell)}(x))$. □

So far we have shown that for every $x \in X$ we have a monotonically increasing sequence $(w(p^{(\ell)}(x)))_{\ell\in\mathbb{N}}$, which is bounded by $V(x)$ due to Proposition 3.1. The following theorem states that the limit is indeed $V(x)$.

Theorem 3.3. *The approximations $V_{\mathcal{P}^{(\ell)}}$ converge pointwise to the value function V as $\ell \to \infty$, that is, for every $x \in X$*

$$V_{\mathcal{P}^{(\ell)}}(x) \to V(x) \quad \text{as } \ell \to \infty.$$

The essential problem in proving this theorem lies in the fact that a minimizing path $p^{(\ell)}(x) = (e_0^{(\ell)}, \dots, e_{m(\ell)}^{(\ell)})$ does not necessarily imply the existence of a control sequence $\mathbf{u}$ and a trajectory

$(x_k(x,\mathbf{u}))_{k\in\mathbb{N}}$ such that $x_k(x,\mathbf{u}) \in P_k^{(\ell)}$ for all $k = 0,\dots,m(\ell)$. Therefore, let us first show that such a control sequence indeed exists in the limit as $\ell \to \infty$.

In order to do so, we wish to work with paths $p^{(\ell)}(x) = (e_0^{(\ell)}, e_1^{(\ell)}, \dots)$ consisting of infinitely many edges. Note that $f(0,0) = 0$, so for every $\ell \in \mathbb{N}$ there is an edge $o^{(\ell)} = (P^{(\ell)}(0), P^{(\ell)}(0))$. Furthermore, since $q(0,0) = 0$, its weight is $w(o^{(\ell)}) = 0$. So formally, we can extend the path $p^{(\ell)}(x)$ by appending the arc $o^{(\ell)}$ infinitely many times, i.e.

$$p^{(\ell)}(x) = (e_k^{(\ell)})_{k\in\mathbb{N}} := (e_1^{(\ell)}, \dots, e_{m(\ell)}^{(\ell)}, o^{(\ell)}, o^{(\ell)}, \dots),$$

while the length of $p^{(\ell)}(x)$ remains the same.

Let us now fix $x \in X$ and consider the infinite sequence of minimizing paths $p^{(\ell)}(x) = (e_1^{(\ell)}, e_2^{(\ell)}, \dots)$ in $G_{\mathcal{P}^{(\ell)}}$ with $\ell \in \mathbb{N}$. Here, $e_k^{(\ell)} = o^{(\ell)}$ for $k > m(\ell)$ and $e_k^{(\ell)} = (P_{k-1}^{(\ell)}, P_k^{(\ell)})$ otherwise. We shall construct an optimal control strategy $\mathbf{u} = (u_0, u_1, \dots) \in \mathcal{U}(x)$ and the associated optimal trajectory $(x_k(x,\mathbf{u}))_{k\in\mathbb{N}}$ such that $p^{(\ell)}(x)$ indeed corresponds to this trajectory in the limit as $\ell \to \infty$. Note that, by definition, $x_0 = x \in P_0^{(\ell)}$, for all $\ell \in \mathbb{N}$.

Let us start by constructing the next point x_1 of the desired trajectory $(x_k(x,\mathbf{u}))_{k\in\mathbb{N}}$ together with a corresponding control u_0. For every ℓ choose a point $x_1^{(\ell)} \in P_1^{(\ell)}$. Since X is compact, the sequence $(x_1^{(\ell)})_{\ell\in\mathbb{N}}$ has a convergent subsequence $(x_1^{(\ell)})_{\ell\in\hat{L}_0}$, for some $\hat{L}_0 \subset \mathbb{N}$. We denote its limit by x_1.

Proposition 3.4. *There is a control $u_0 \in \mathbb{R}^m$ such that $x_1 = f(x_0, u_0)$.*

Proof. By construction of the graphs $G_{\mathcal{P}^{(\ell)}}$, for every $\ell \in \hat{L}_0$, there is a point $x_0^{(\ell)} \in P_0^{(\ell)}$ and a control $u_0^{(\ell)} \in \mathbb{R}^m$ such that $x_1^{(\ell)} = f(x_0^{(\ell)}, u_0^{(\ell)})$. Since $\operatorname{diam}(\mathcal{P}^{(\ell)}) \to 0$ as $\ell \to \infty$, we have

$$\lim_{\ell\in\hat{L}_0} x_0^{(\ell)} = x_0 \quad \text{and} \quad \lim_{\ell\in\hat{L}_0} x_1^{(\ell)} = x_1.$$

Since U is compact, we can choose a convergent subsequence $(u_0^{(\ell)})_{\ell\in L_0}$ of $(u_0^{(\ell)})_{\ell\in\hat{L}_0}$, with $L_0 \subset \hat{L}_0$. We denote its limit by u_0. By continuity of f we must have $x_1 = f(x_0, u_0)$. □

We can continue our construction of the trajectory $(x_k(x,\mathbf{u}))_{k\in\mathbb{N}}$ by repeating the above with x_1 instead of $x_0 = x$, that is, for $\ell \in L_0$ we choose points $x_2^{(\ell)} \in P_2^{(\ell)}$, a convergent subsequence

$(x_2^{(\ell)})_{\ell\in\hat{L}_1}$, $\hat{L}_1 \subset L_0$, with limit x_2 and a corresponding control $u_1 \in U$ as the limit of a convergent subsequence $(u_1^{(\ell)})_{\ell\in L_1}$, $L_1 \subset \hat{L}_1$, such that $x_2 = f(x_1, u_1)$, etc.

The construction of the desired trajectory $(x_k(x, \mathbf{u}))_{k\in\mathbb{N}}$ can now be done using a standard "diagonal argument": Note that we implicitly constructed a sequence $(L_k)_{k\geq 0}$ of subsets of $\mathbb{N}$, such that $L_{k+1} \subset L_k$ for all $k \geq 0$. Choose some arbitrary element $\ell_0 \in L_0$ and inductively let $\ell_k \in L_k, k = 1, 2, \ldots$ be the smallest number such that $\ell_k > \ell_{k-1}$. Define $L = \{\ell_0, \ell_1, \ldots\}$. By construction, for every k, the subset $\{\ell_k, \ell_{k+1}, \ldots\}$ of L is contained in L_k. Furthermore, for all $k = 0, 1, \ldots$

$$\lim_{\ell\in L} x_k^{(\ell)} = \lim_{\ell\in L_k} x_k^{(\ell)} = x_k \quad \text{and} \quad \lim_{\ell\in L} u_k^{(\ell)} = \lim_{\ell\in L_k} u_k^{(\ell)} = u_k.$$

The incremental cost of each iterate in the trajectory $(x_k(x, \mathbf{u}))_{k\in\mathbb{N}}$, with $\mathbf{u} = (u_0, u_1, \ldots)$, is now directly related to the limits of the weights of the associated edges in $G_{\mathcal{P}^{(\ell)}}$. Namely, for $k = 0, 1, \ldots,$

$$q(x_k, u_k) = \lim_{\ell\in L} q(x_k^{(\ell)}, u_k^{(\ell)}) = \lim_{\ell\in L} w(e_{k+1}^{(\ell)}).$$

We are now ready to proof Theorem 3.3.

Proof of Theorem 3.3. Suppose that for some $x \in X$ the statement does not hold, that is,

$$\lim_{\ell\to\infty} V_{\mathcal{P}^{(\ell)}}(x) = \bar{V}(x) < V(x).$$

We are going to show the existence of a controlled trajectory with cost $J(x, \mathbf{u}) \leq \bar{V}(x)$. This violates the definition of V. Following the construction above we obtain a controlled trajectory $(x_k(x, \mathbf{u}))_{k\in\mathbb{N}}$ with total cost

$$\begin{aligned} J(x, \mathbf{u}) = \sum_{k=0}^{\infty} q(x_k, u_k) &= \lim_{K\to\infty} \sum_{k=0}^{K} \lim_{\ell\in L} q(x_k^{(\ell)}, u_k^{(\ell)}) \\ &= \lim_{K\to\infty} \sum_{k=0}^{K} \lim_{\ell\in L} w(e_{k+1}^{(\ell)}) = \lim_{K\to\infty} \lim_{\ell\in L} \sum_{k=0}^{K} w(e_{k+1}^{(\ell)}). \end{aligned}$$

Since $w(e_{k+1}^{(\ell)}) = 0$ for all $k > m(\ell)$, we can stop the summation at $m(\ell)$, possibly increasing the sum when $m(\ell) > K$. Hence, we obtain

$$J(x, \mathbf{u}) \leq \lim_{\ell \in L} \sum_{k=0}^{m(\ell)} w(e_{k+1}^{(\ell)}) = \lim_{\ell \in L} V_{\mathcal{P}^{(\ell)}}(x) = \bar{V}(x).$$

This yields the desired contradiction and completes the proof of Theorem 3.3. □

4. Implementation

The central computational step is the construction of the finite graph

$$G_{\mathcal{P}} = (\mathcal{P}, E_{\mathcal{P}}), \quad E_{\mathcal{P}} = \{(P_i, P_j) \in \mathcal{P} \times \mathcal{P} \mid f(P_i, U) \cap P_j \neq \emptyset\}, \tag{4}$$

in particular determining the edges $e = (P_i, P_j) \in E_{\mathcal{P}}$ with their associated weights

$$w(e) = \min_{x \in P_i, u \in U} \{q(x, u) \mid f(x, u) \in P_j\}. \tag{5}$$

Once this weighted graph has been computed, standard algorithms from graph theory (for example, "all source, single destination" shortest path algorithms like Dijkstra's algorithm [2,5]) can be applied in order to compute the approximate value function $V_{\mathcal{P}}$ in (3).

Essentially, the construction of $G_{\mathcal{P}}$ breaks down into the following three steps

(1) Construction of a suitable partition $\mathcal{P}$ of the region of interest $X \in \mathbb{R}^d$;
(2) Construction of the set $E_{\mathcal{P}}$ of edges of $G_{\mathcal{P}}$;
(3) Computation of the weights $w(e)$ for the edges $e \in E_{\mathcal{P}}$.

One can think of the first two steps as the problem of constructing the graph $G_{\mathcal{P}}$ from a given (nonlinear) control problem. The last step, that is, the weights involve the cost function and make the graph relevant for solving an optimal control problem.

It is the interlocking of these three steps in a multilevel approach, as pioneered in [3], that makes our method computationally efficient. Namely, the construction of $G_{\mathcal{P}}$ does not happen step by step, determining first $\mathcal{P}$, then $E_{\mathcal{P}}$ and only then $w(e)$ for $e \in E_{\mathcal{P}}$. Instead, we build $G_{\mathcal{P}}$ using a nested sequence $G_{\mathcal{P}^{(\ell)}}$ of partitions. The central idea is a hierarchical multilevel approach to the construction and storage of $\mathcal{P}$. This has the following two advantages:

- Possibly large regions of X are removed from consideration at early stages in the procedure. Namely, these regions are not contained in S and a finer partition for such regions is not necessary.
- The complexity of the computation of the set $E_{\mathcal{P}}$ of edges of $G_{\mathcal{P}}$ reduces from $\mathcal{O}(N^2)$, using a naive approach, to at least $\mathcal{O}(N \log N)$ (where N is the number of elements in $\mathcal{P}$).

For an in-depth description of this approach see [4, 3, 17]. It is implemented in the software package GAIO[1], which was also used for the computations in Section 5.

Note that it is essential to restrict to a compact subset $X \in \mathbb{R}^d$ and it is not possible to include trajectories that are not entirely confined to X. For problems with non-compact phase spaces, for example, the whole of $\mathbb{R}^d$, we can only find an approximation of the value function on a compact subset X. However, in this case $V_{\mathcal{P}}$ will not be a lower bound everywhere and regions may be deemed not stabilizable, because they are not stabilizable in X.

4.1. **Construction of the partitions**

We are going to use d-dimensional boxes as the elements of a partition $\mathcal{P}$. These boxes are stored in a binary tree which builds up a nested sequence of partitions $\mathcal{P}^{(\ell)}$ up to a required level. In summary, starting with a coarse partition $\mathcal{P}^{(\ell)}$, a finer partition $\mathcal{P}^{(\ell+1)}$ is constructed by bisecting the boxes in $\mathcal{P}^{(\ell)}$ with respect to some coordinate direction. At first this doubles the number of boxes, however, once we computed $E_{\mathcal{P}^{(\ell+1)}}$ we will be removing some of them.

4.2. **Construction of the edges**

The set of edges $E_{\mathcal{P}^{(\ell+1)}}$ can then be constructed using the information of $E_{\mathcal{P}^{(\ell)}}$. For every box $P_i^{(\ell)} \in \mathcal{P}^{(\ell)}$ there are at first two boxes $P_{i,-}^{(\ell+1)}, P_{i,+}^{(\ell+1)} \in \mathcal{P}^{(\ell+1)}$ with $P_i^{(\ell)} = P_{i,-}^{(\ell+1)} \cup P_{i,+}^{(\ell+1)}$. Hence, only if there is an edge $e^{(\ell)} = (P_i^{(\ell)}, P_j^{(\ell)})$ in $E_{\mathcal{P}^{(\ell)}}$, one needs to check whether there should be edges $e^{(\ell+1)} = (P_{i,\sigma}^{(\ell+1)}, P_{j,\tau}^{(\ell+1)})$ in $E_{\mathcal{P}^{(\ell+1)}}$, $\sigma, \tau \in \{-, +\}$. This check requires to determine whether

$$f(P_{i,\sigma}^{(\ell+1)}, U) \cap P_{j,\tau}^{(\ell+1)} \neq \emptyset \tag{6}$$

[1] `http://math-www.upb.de/~agdellnitz/gaio`

for $\sigma, \tau \in \{-, +\}$. Typically, this check is performed using a finite number of *test points* $(x, u) \in T_{i,\sigma}^{(\ell+1)} \subset P_{i,\sigma}^{(\ell+1)} \times U$. Obviously, by discretizing (6) using arbitrary test points, one introduces the possibility that not all edges are detected. When certain Lipschitz-constants on f are known, this discretization can be made rigorous [12]. In some cases it is also possible to use interval arithmetic in order to check condition (6); see e.g. [7].

In the examples of Section 5 we used test points, but ignored the special relation between edges of level $\ell + 1$ and edges of level ℓ. Instead, we considered *all* boxes at level $\ell + 1$ when determining which element of $\mathcal{P}^{(\ell+1)}$ contains the image of a test point (x, u). This technique renders the computation of the edges more robust. Furthermore, due to the hierachical storage of the collections $\mathcal{P}^{(\ell)}$, the complexity of this search is still only $\mathcal{O}(\log N)$ (where N is the number of elements in $\mathcal{P}^{(\ell+1)}$).

4.3. **Computation of the weights**

Once the graph $G_{\mathcal{P}}$ as defined in (4) has been computed, the computation of the weights (5) reduces to a (nonlinear) optimization problem. This can in principle be solved by standard methods (see e.g. [9]). In the examples we worked with a rough approximation only, using

$$w(e) \approx \min_{(x,u)\in T_i} \{q(x, u) \mid f(x, u) \in P_j\}, \quad e = (P_i, P_j) \in E_{\mathcal{P}},$$

with $T_i \subset P_i \times U$ the finite set of test points.

Remark 4.1. If we could determine the edges $E_{\mathcal{P}}$ and associated weights exactly, Proposition 3.1 would hold. However, with the use of test points, we introduced the possibility of errors that lead to an approximation $V_{\mathcal{P}}$ of V that may no longer be a lower bound. Similarly Proposition 3.2 could fail when $T_{i,-}^{(\ell+1)} \cup T_{i,+}^{(\ell+1)} \supset T_i^{(\ell)}$, which is to be expected, because one would typically choose a fixed number of test points regardless of the diameter of the box. Nevertheless, in practice, $V_{\mathcal{P}}$ seems to be a good approximation of V.

5. Examples

To demonstrate the effectiveness of our approach, we consider the problems of balancing single and double inverted pendula. We find that our method is very good to get a rough idea of the

behaviour of the value function, because the computations are extremely fast. Furthermore, even with a crude partition and very few test points, the results appear to be quite accurate. A detailed error analysis of the method will be the topic of future investigations.

5.1. **Single inverted pendulum on a cart**

As a first example, we consider balancing a planar inverted pendulum on a cart that moves under an applied horizontal force u, constituting the control, in a direction that lies in the plane of motion of the pendulum. This problem was also studied in [10, 11] and we can compare the results.

The position of the pendulum is measured relative to the position of the cart as the offset angle φ from the vertical up position. We completely ignore the dynamics of the cart and focus on the two-dimensional state space $(\varphi, \dot{\varphi}) \in \mathbb{R}^2$ describing the motion of the pendulum. Assuming that there is no friction, the equations of motion can be derived from first principles. Here, we use $M = 8\,\mathrm{kg}$ for the mass of the cart, $m = 2\,\mathrm{kg}$ for the mass of the pendulum. The center of mass lies at distance $l = 0.5\,\mathrm{m}$ from the pivot. Writing $x_1 = \varphi$ and $x_2 = \dot{\varphi}$, the equations become

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{\frac{g}{l}\,\sin(x_1) - \frac{1}{2}\,m_r\,x_2^2\,\sin(2x_1) - \frac{m_r}{ml}\,\cos(x_1)\,u}{\frac{4}{3} - m_r\,\cos^2(x_1)} \end{aligned} \tag{7}$$

where $m_r = m/(m+M)$ is the mass ratio. We use $g = 9.8\mathrm{m/s}^2$ for the gravitational constant. The stabilization of this inverted pendulum is subject to the incremental cost

$$q(x,u) = \frac{1}{2}(0.1x_1^2 + 0.05x_2^2 + 0.01u^2). \tag{8}$$

These parameters are as in [10] where the value function was computed using a completely different method.

For our computations, we need to obtain a discrete-time control system. To this end, we consider the time-T map

$$f(x,u) = \phi^T(x;u), \tag{9}$$

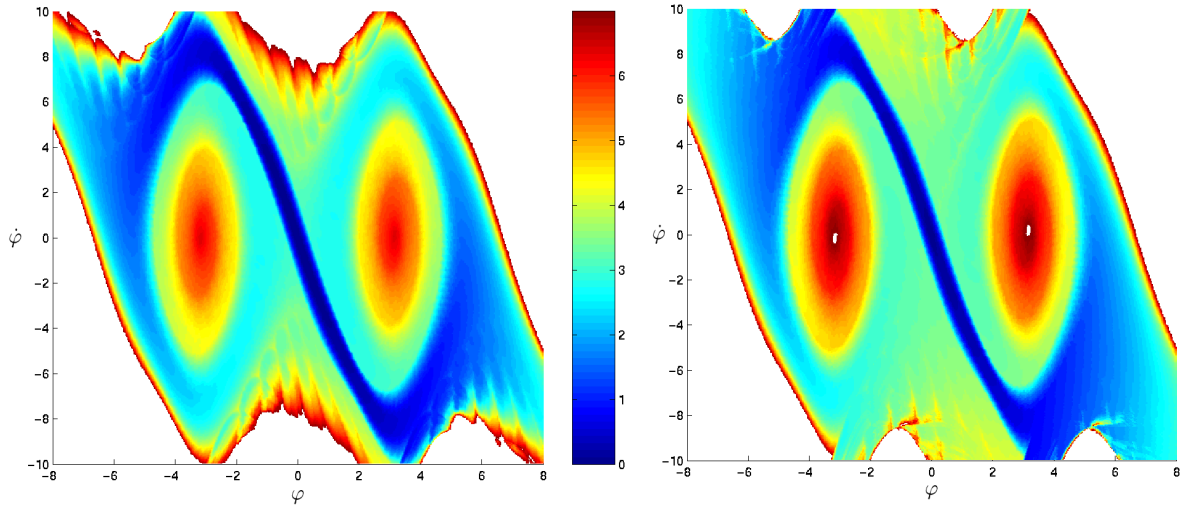

FIGURE 1. The approximate value function $V_{\mathcal{P}}$ for the single inverted pendulum on a cart. The size of the boxes in the partition $\mathcal{P}$ is $1/64 \times 5/256$. The region of interest is $X = [-8, 8] \times [-10, 10]$ and the controls are restricted to $U = [-64, 64]$ left and $U = [-128, 128]$ right, respectively. The optimal cost for each initial condition is represented by a color from 0 (blue) to red (7).

where ϕ^t is the flow associated with Equation (7), T is some fixed time and $u \equiv u(t)$ is the constant control action. The instantaneous cost function of the time-T map is

$$q_T(x, u) = \int_0^T q(\phi^t(x; u), u)\, dt, \tag{10}$$

where, again, u denotes the constant function with value u on $[0, T]$.) For this example we used $T = 0.1$ and each iterate was computed via the Runge-Kutta scheme of 4-th order with step size 0.02. As in [10], we choose $X = [-8, 8] \times [-10, 10]$ as the region of interest . We used a partition $\mathcal{P}$ of this region with boxes of size $1/64 \times 5/256$. The computation of the weights was based on a total of 4 equally spaced test points per box (i.e. the vertices) in phase space and 20 equally spaced points in control space.

Let us first illustrate the effect of restricting the controls to a bounded set. Figure 1 shows two approximations of the value function V. In both pictures, each box $P_i \in \mathcal{P}$ has been given a color representing the approximate optimal cost of driving an initial condition $x \in P_i$ to the origin. Cost runs from 0 (blue) to 7 (red). The upper bound of 7 is artificial, and white regions corresponds to points that either have a higher optimal cost, or are not stabilizable and

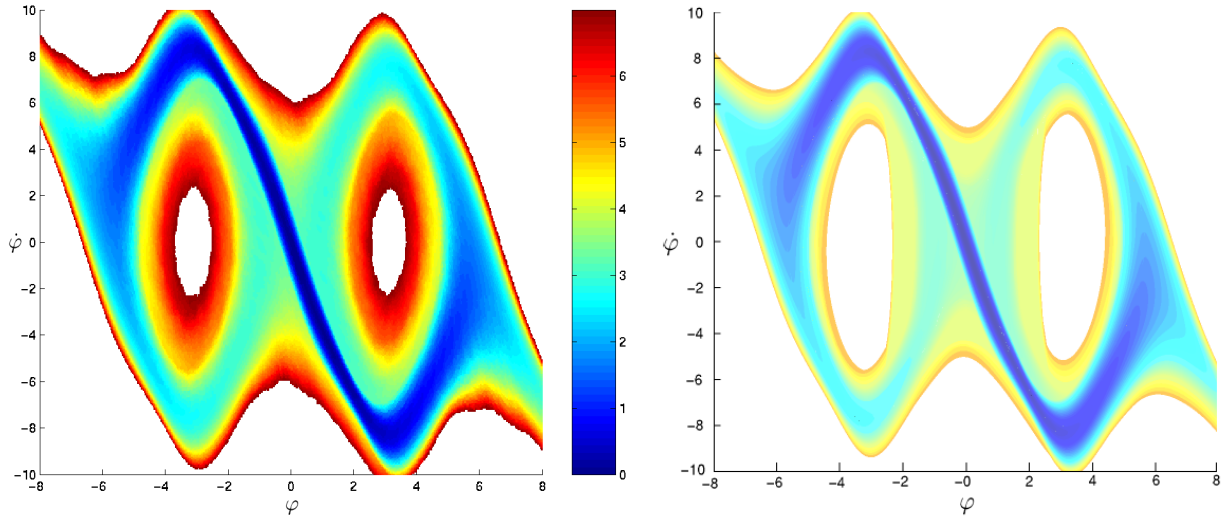

FIGURE 2. Left: approximate value function $V_{\mathcal{P}}$ for the single inverted pendulum on a cart. The size of the boxes in the partition $\mathcal{P}$ is $2^{-5}\times 2^{-4}$. Right: approximate value function as computed in [10]. Again, the optimal cost for each initial condition is represented by a color from 0 (blue) to red (7).

the approximate value function is infinite. For the left illustration of Figure 1 the controls are restrictred to $U = [-64, 64]$, and for the right to $U = [-128, 128]$. The pictures agree quite well for relatively small (in absolute value) velocities $\dot{\varphi}$. However, for relatively large values of $|\dot{\varphi}|$, especially near $|\varphi| = 0$ the optimal cost is a lot higher when the controls are restricted more severely.

The illustrations in Figure 1 also show another interesting feature. Along the boundary of the colored region, the approximate value function exhibits "oscillations". This behavior is most pronounced in the left picture. The effect that we see here is due to the restriction that an optimal trajectory must lie entirely in X. The bubbles with lower optimal cost are associated with optimal trajectories that are not affected by this restriction, that is, the true optimal trajectory is contained in X. If we enlarge X this effect will again be pushed to the boundary of the region of interest.

In order to illustrate the quality of the approximation, we compared our computations with those of [10]. To this end we computed an approximation $V_{\mathcal{P}}$ of the value function V using a partition $\mathcal{P}$ with boxes of size $2^{-5} \times 2^{-4}$ of the region $X = [-8, 8] \times [-16, 16]$, while the

actual region of interest (the plotted region) remained $[-8,8]\times[-10,10]$. (It is indeed enough to only enlarge the region for $\dot\varphi$.) This results in a disappearance of the "oscillations" near the boundary. This time we used an integration time of $T=0.4$, realized by ten Runge-Kutta steps. The computation of the weights was again based on a total of 4 equally spaced test points per box in phase space and 20 equally spaced points in control space. The set of admissible controls was chosen as $U=[-120,120]$. This is in accordance with the computations in [10] where the absolute value of the controls never exceeded 120. The approximation $V_{\mathcal{P}}$ is visualized in Figure 2 (left) using the same color scheme as Figure 1.

The computations in [10] were done using a different method and the value function is represented as a collection of *optimal cost isoclines*. The computations were done up to level $V=\frac{1}{2}3.3^2=5.445$. The visualization of this data is shown in Figure 2(right). Note that the same color scheme as Figure 2(left) was used, with a maximal cost of 7, so red is missing in this picture. We remark that a similar picture can be found in [10, Fig. 2], where the coloring is done proportional to the square root of the cost (in fact, the factor $\frac{1}{2}$ in the cost Equation (8) is missing in this figure). It should be noted that the computations of [10] are extremely accurate, but their method is very cumbersome and the computation of the optimal cost isoclines up to V = 5.445 took about one week.

Our computation for $V_{\mathcal{P}}$ took only a couple of minutes on a recent workstation. Even when using a fairly crude partition and only few test points the comparison is already quite good. However, one can see that the "numerical" $V_{\mathcal{P}}$ is not a lower bound of V, as predicted by the theory. We believe that this is (partly) due to the fact that we compute the weights from the test points, assigning too high an optimal cost to each edge.

5.2. **Parametrically forced inverted double pendulum**

Our method has the nice property that it is readily applicable to higher-dimensional spaces. As an example we consider the problem of balancing a parametrically driven (planar) inverted double pendulum [14]. The phase space $x=(\varphi_1,\dot\varphi_1,\varphi_2,\dot\varphi_2)\in\mathbb{R}^4$ is four-dimensional. The

equations of motion are

$$\begin{cases} 2\ddot{\varphi}_1 + \ddot{\varphi}_2 \cos(\varphi_1 - \varphi_2) &= 2\left(\dfrac{g+u}{\ell}\right)\sin\varphi_1 - \dot{\varphi}_2^2 \sin(\varphi_1 - \varphi_2), \\ \ddot{\varphi}_2 + \ddot{\varphi}_1 \cos(\varphi_1 - \varphi_2) &= \left(\dfrac{g+u}{\ell}\right)\sin\varphi_2 + \dot{\varphi}_1^2 \sin(\varphi_1 - \varphi_2), \end{cases} \tag{11}$$

where $\ell = 0.5$. As the corresponding discrete-time system we consider the time-T map with $T = 0.4$. The integration is done using the 4-th order Runge-Kutta scheme with constant step size 0.04. The cost function is defined as in (10) with

$$q(x,u) = \frac{1}{2}(0.1\,\varphi_1^2 + 0.05\,\dot{\varphi}_1^2 + 0.1\,\varphi_2^2 + 0.05\,\dot{\varphi}_2^2 + 0.01\,u^2).$$

We use $X = [-2\pi, 2\pi] \times [-4\pi, 4\pi] \times [-2\pi, 2\pi] \times [-4\pi, 4\pi]$ as the region of interest, and $U = [-4, 4]$ for the control space. The computations are done using a rather crude partition of X into boxes of radii $(\frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{8}, \frac{\pi}{4})$. We use 81 test points (on a regular grid) in each box in phase space, and 9 equally spaced points in control space.

Unfortunately, it is not possible to visualize the value function as in the previous example, because this data set is four-dimensional. However, one can show an approximate optimal trajectory for just one arbitrarily chosen initial condition. For example, Figure 3 shows such an approximation for the point $x_0 = (\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2) = (\pi, 0, \pi, 0)$. In order to obtain the approximate optimal trajectory we consider the shortest path $p = p(x_0)$ connecting a box in the partition that contains x_0 to a box that contains the origin. By definition, each edge $e = (P_i, P_j)$ in $p(x_0)$ is associated with a particular state $x_e = (\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2) \in P_i$ and control $u_e \in U$ such that

$$w(e) = \min_{x \in P_i, u \in U} \{q(x,u) \mid f(x,u) \in P_j\} = q(x_e, u_e).$$

The approximate optimal trajectory in Figure 3 is the piecewise continuous concatenation of trajectories obtained by integrating (11) for time $T = 0.4$ with initial condition x_e and control $u \equiv u_e$, where x_e and u_e are associated with edges along the shortest path $p(x_0)$. For example, the first edge of the shortest path starting at a box containing $(\pi, 0, \pi, 0)$ is associated with the point $x_e = (\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2) \approx (2.37, -0.78, 3.15, -0.78)$ and uses the control $u_e = -1$.

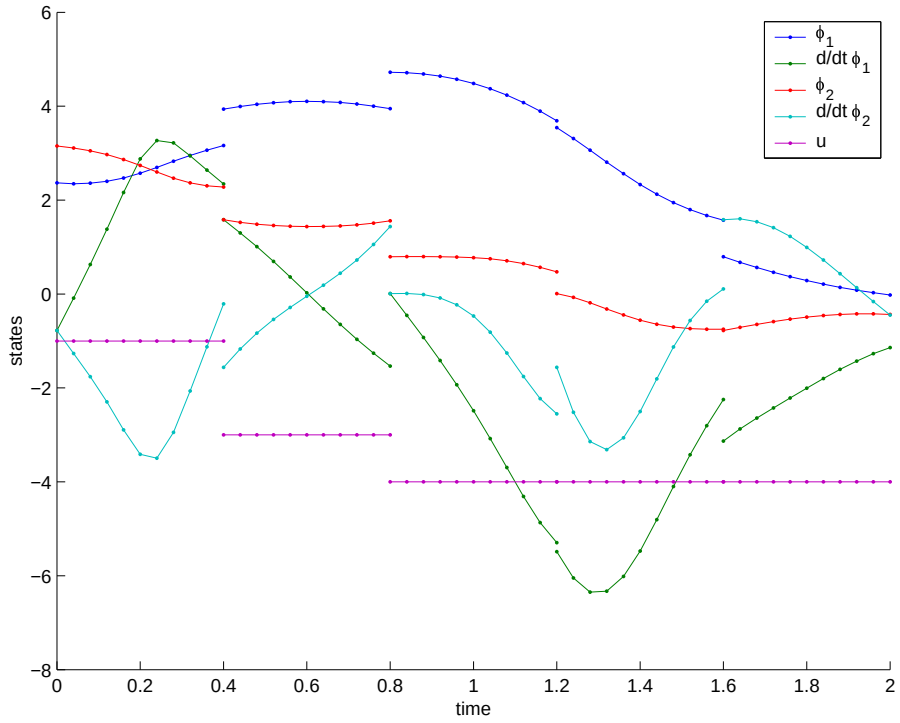

FIGURE 3. Approximate optimal pseudo-trajectory starting at $x_e = (\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2) \approx (2.37, -0.78, 3.15, -0.78)$, close to the vertical down position, that drives the double inverted pendulum close to the upright position.

Figure 3 shows all variables $\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2$ plotted versus time. The approximate piecewise constant optimal control action u is shown as well. For each variable we also plotted the integration steps used in the numerical integration of the time-T-map of (11).

Due to the nature of our computations, the approximate optimal trajectory is only a piecewise continuous curve with discontinuities at each time-T iterate. Note that such discontinuities are to be expected, because the initial condition for each iterate is determined by the weight of the edge in the optimal path, rather than by the endpoint of the previous iteration. As mentioned, this approximate optimal "pseudo-trajectory" should be viewed as an intial guess for standard (local) solvers for optimal control problems (see e.g. [19]).

As a further analysis of the four-dimensional data set we extract a two-dimensional domain by considering initial conditions on the "diagonal" $\{\varphi_1 = \varphi_2, \dot{\varphi}_1 = \dot{\varphi}_2\}$. This subset is the set of initial conditions where the two pendula are aligned and moving with the same angular velocity.

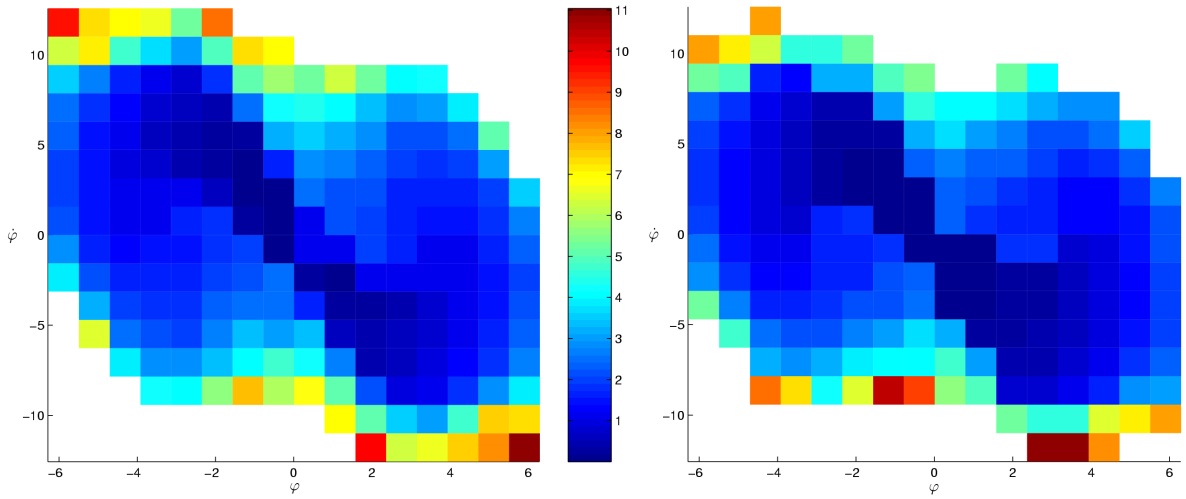

FIGURE 4. Left: the approximate value function $V_{\mathcal{P}}$ for the parametrically forced inverted double pendulum on the subset $\{\varphi = \varphi_1 = \varphi_2, \dot{\varphi} = \dot{\varphi}_1 = \dot{\varphi}_2\}$. Right: the approximate value function $V_{\mathcal{P}}$ for the parametrically forced inverted single pendulum. The radii of the boxes in the partition is $(\frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{8}, \frac{\pi}{4})$. The controls are restricted to $U = [-4, 4]$. The optimal cost for each initial condition is represented by a color. Blue represents low (0) and red high (11) cost.

Hence, at least initially, one would expect the same behavior as for a single parametrically forced inverted pendulum. The approximate value function of this diagonal subset for the double pendulum is shown in Figure 4(left).

We also computed the approximate value function of the parametrically forced inverted single pendulum

$$\ddot{\varphi} - \left(\frac{g+u}{l}\right)\sin(\varphi) = 0, \tag{12}$$

where $l = 1$ in this case and the cost function is chosen such that it matches the cost function for the double pendulum restricted to the set $\{\varphi_1 = \varphi_2, \dot{\varphi}_1 = \dot{\varphi}_2\}$, that is,

$$q(x, u) = \frac{1}{2}(0.2\varphi^2 + 0.1\dot{\varphi}^2 + 0.01u^2).$$

Here, $x = (\varphi, \dot{\varphi}) \in \mathbb{R}^2$. We used the same discretization as for the double pendulum and concentrated on the region $X = [-2\pi, 2\pi] \times [-4\pi, 4\pi]$ with $U = [-4, 4]$. We used a similarly crude partition with boxes of radii $(\frac{\pi}{8}, \frac{\pi}{4})$ with 9 test point in phase space and 9 in control

space. The approximate value function of the parameterically forced inverted single pendulum is shown in Figure 4(right).

References

[1] R.A. Brooks and T. Lozano-Pérez, A subdivision algorithm in configuration space for findpath with rotation. *IEEE Systems, Man and Cybernetics* **SMC-15** (1985) 224-233.

[2] T.H. Cormen, C.E. Leierson and R.L. Rivest, *Introduction to Algorithms*. Cambridge, Mass. MIT Press, New York McGraw-Hill (1990).

[3] M. Dellnitz and A. Hohmann, A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik* **75** (1997) 293-317.

[4] M. Dellnitz, G. Froyland and O. Junge, The algorithms behind GAIO — Set oriented numerical methods for dynamical systems, in *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*, edited by B. Fiedler, Springer (2001) 145-174.

[5] E.W. Dijkstra, A Note on Two Problems in Connection with Graphs. *Numerische Mathematik* **5** (1959) 269-271.

[6] M. Falcone, Numerical solution of Dynamic Programming equations, in *Viscosity solutions and deterministic optimal control problems*, edited by M. Bardi and I. Capuzzo Dolcetta, Birkhäuser (1997).

[7] Z. Galias, Interval methods for rigorous investigations of periodic orbits. *International Journal of Bifurcation and Chaos* **11**(9) (2001) 2427-2450.

[8] L. Grüne, An Adaptive Grid Scheme for the discrete Hamilton-Jacobi-Bellman Equation. *Numerische Mathematik* **75** (1997) 319-337.

[9] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, *User's Guide for NPSOL (Version 4.0): a Fortran package for nonlinear programming*, Report SOL 86-2, Systems Optimization Laboratory, Stanford University (1986).

[10] J. Hauser and H.M. Osinga, On the geometry of optimal control: the inverted pendulum example, in *Proc. Amer. Control Conf.*, Arlington VA (2001) 1721-1726.

[11] A. Jadbabaie, J. Yu and J. Hauser, Unconstrained receding horizon control: Stability region of attraction results, in *Proc. Conf. on Decision and Control*,# CDC99-REG0545 (1999).

[12] O. Junge, Rigorous discretization of subdivision techniques, in *Proc. Int. Conf. Differential Equations Equadiff 99*, edited by B. Fiedler, K. Gröger and J. Sprekels, World Scientific **2** (2000) 916-918.

[13] L.C. Polymenakos, D.P. Bertsekas and J.N. Tsitsiklis, Implementation of efficient algorithms for globally optimal trajectories, *IEEE Transactions on Automatic Control* **43**(2) (1998) 278-283.

[14] K. Schiele, On the stabilization of a parametrically driven inverted double pendulum. *Zeitschrift für angewandte Mathematik und Mechanik* **77**(2) (1997) 143-146.

[15] J.A. Sethian and A. Vladimirsky, Ordered upwind methods for static Hamilton-Jacobi equations, *Proceedings of the National Academy of Sciences of the USA* **98**(20) (2001) 11069-11074.

[16] E.D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Texts in Applied Mathematics 6, Springer (1998).

[17] D. Szolnoki, Viability kernels and control sets. *ESAIM: Control, Optimisation and Calculus of Variations* **5** (2000) 175-185.

[18] J.N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, *IEEE Transactions on Automatic Control* **40**(9) (1995) 1528-1538.

[19] O. von Stryk, *User's Guide for DIRCOL (Version 2.1): a direct collocation method for the numerical solution of optimal control problems*, TU Darmstadt (2000).